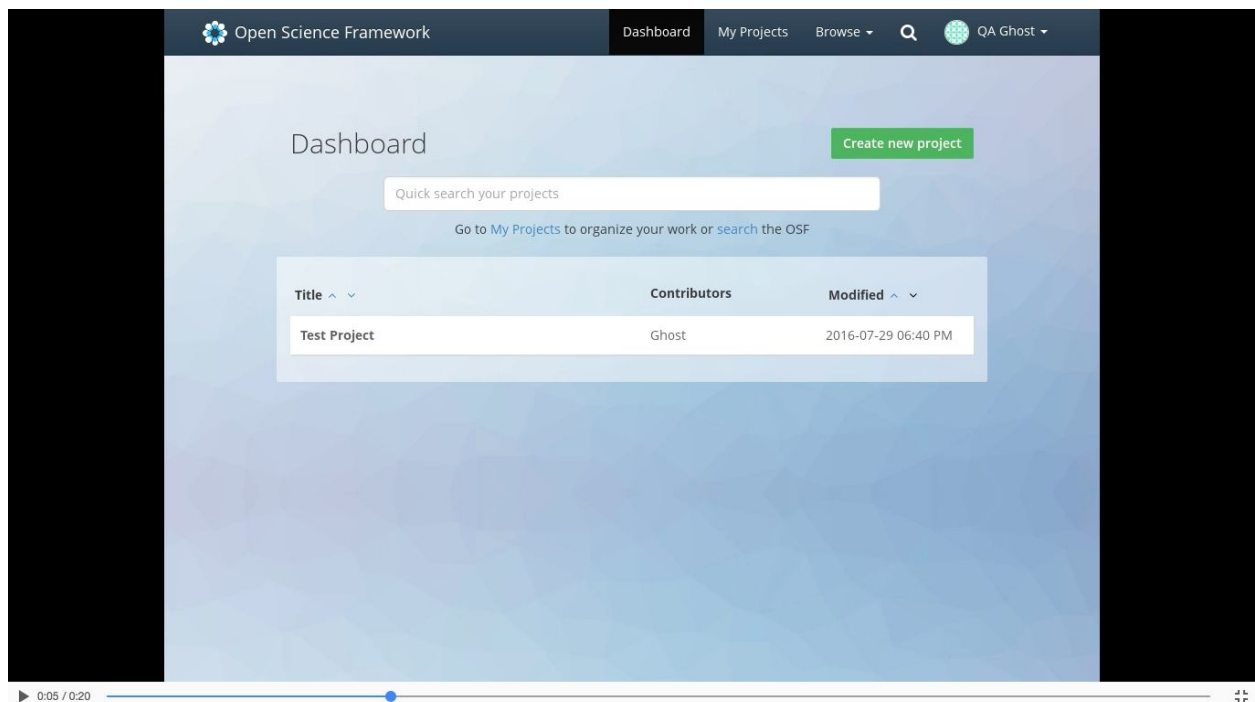


Ghost Inspector Tutorial

Introduction

Welcome to the Ghost Inspector tutorial. This tutorial is designed to give you a basic understanding of how to use Ghost Inspector. Granted, there are a lot of tricky aspects of this tool. With a little practice, you will hopefully obtain an even greater understanding. So don't worry if it all seems confusing at first.

What is Ghost Inspector? Ghost Inspector is a tool that we will use for automated testing, specifically having to do with the User Interface. Think of it like this: with every release, we want to make sure that our website is bug-free and working exactly how we expect it to work. Normally, we would have to spend a large portion of time clicking around, making projects, adding files, deleting contributors, checking to see if everything looks good, whatever. All manually, as in *literally by hand*. (Or keyboard if you want to get really specific.) Ghost Inspector allows us to make tests that will run automatically and return results in the form of a “pass” or a “fail”, along with a video to see the process ourselves.



Setting Up

First, you will need to create an account. [Here's](#) the link. Then you will need to ask someone to have you added to the “Center for Open Science” group within Ghost Inspector so you can see all of the current tests. Ask me ([Patrick](#)) or anyone else on the QA team for help with this.

Having been successfully added, you should see a collection of test suites. There are only two test suites that we will actually be using: “QA - Production”, and “QA - Staging”.

The screenshot displays the Ghost Inspector interface for the 'Center for Open Science' organization. It lists several test suites with their status and test counts. Two suites, 'QA - Production' and 'QA - Staging', are circled in red. A sidebar on the right shows the 'Create New Suite' form.

SUITE NAME	SCHEDULE	TESTS
.81 Release 2 Passing – Steps, 1 Passing – Screenshot	No Schedule	2
P1: OSF Site and Project Functionality 7 Passing – Steps, 1 Passing – Screenshot, 6 Failing – Steps	No Schedule	13
QA - Production 9 Passing – Steps This is the bucket for all of our production regression tests. It has been formatted to resemble our QA - Production bucket in Runscope. The numbered tests are complete tests that should be used to check functionality, while the non-numbered tests are built...	No Schedule	9
QA - Staging 9 Passing – Steps This is the bucket for all of our staging tests. It has been formatted to resemble our QA - Staging bucket in Runscope, and to mirror our QA - Production bucket in Ghost Inspector. The numbered tests are complete tests that should be used to check function...	No Schedule	9
Test 1 Passing – Steps, 1 Passing – Screenshot	No Schedule	1

Create New Suite

Organization: Center for Open Science

Suite Name: ex: eCommerce Website Tests

Description: ex: A group of tests to check the functionality of our eCommerce store.

Create Suite

For the most part, these two suites are practically identical.

Production will be used for when we perform our regression tests (the tests we run to make sure everything that looked good on staging also looks good on production). This won't be changed much, aside from copying tests over from Staging. You do not want to be adding tests here that aren't in Staging unless for a very specific reason.

Staging will be used for creating new tests and updating old tests that are in need of modification due to a bug fix or an improvement. Whenever we make a change to the Staging bucket, we will copy over this change to the Production bucket when the next release occurs.

This is the bucket for all of our staging tests. It has been formatted to resemble our QA - Staging bucket in Runscope, and to mirror our QA - Production bucket in Ghost Inspector. The numbered tests are complete tests that should be used to check function... [Show](#)

API Access

Your account includes API access. Below are some links for carrying out various actions. [Full documentation](#)

We have a Google Doc dedicated to brainstorming building block tests and numbered tests. [Here's](#) the link. Tests with (BB) next to them are building block tests. Tests with (#) next to them are numbered tests. They will eventually have links to their respective tests. Some have them already.

Ok. That's enough talk about the administrative tasks involved. Boring, yes. But at the same time highly critical. I mean, you did join the QA team. What'd you expect? ;)

Let's talk about how to actually make a test.

How to Actually Make a Test

In all honesty, playing around with Ghost Inspector is the best way to learn. But this will help you with the fundamentals. You will be using Google Chrome for creating all of your tests, because you want to download Ghost Inspector's Chrome extension [here](#). They have a video tutorial on the page, which you are welcome to watch. What it basically does is allow you to record whatever you click on or fill out automatically, step-by-step.

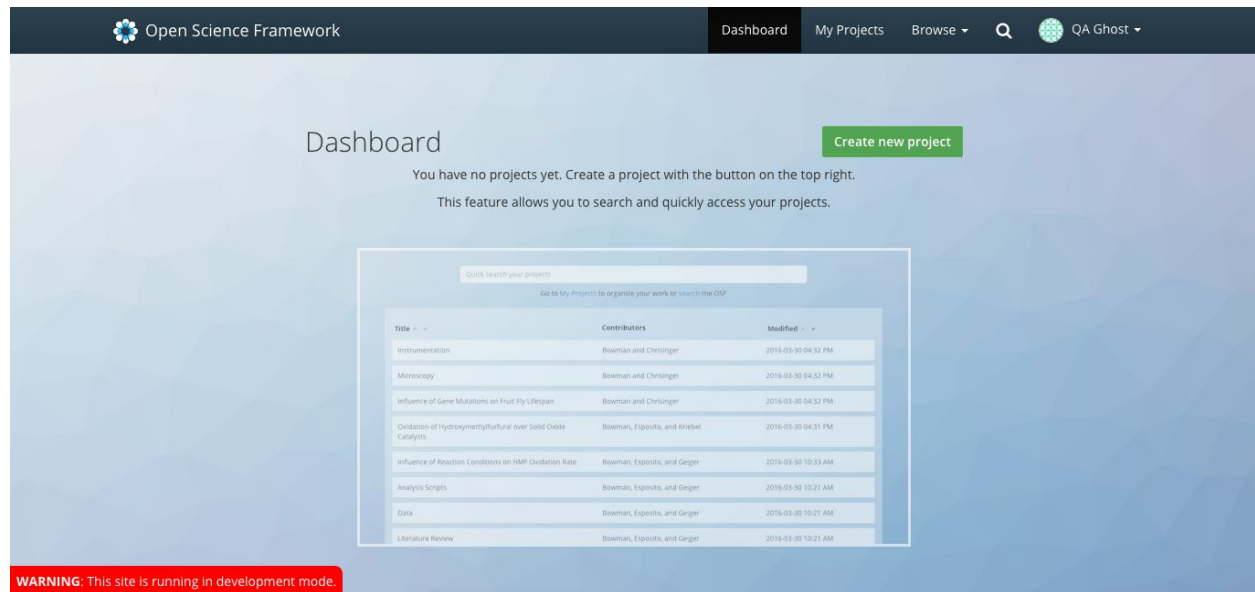
For the purposes of this tutorial, I am going to be going through a relatively basic example that covers almost everything you need to know. We will be creating a project with Ghost Inspector.

First, try to think about what you need to do when you create a project. The steps involved.

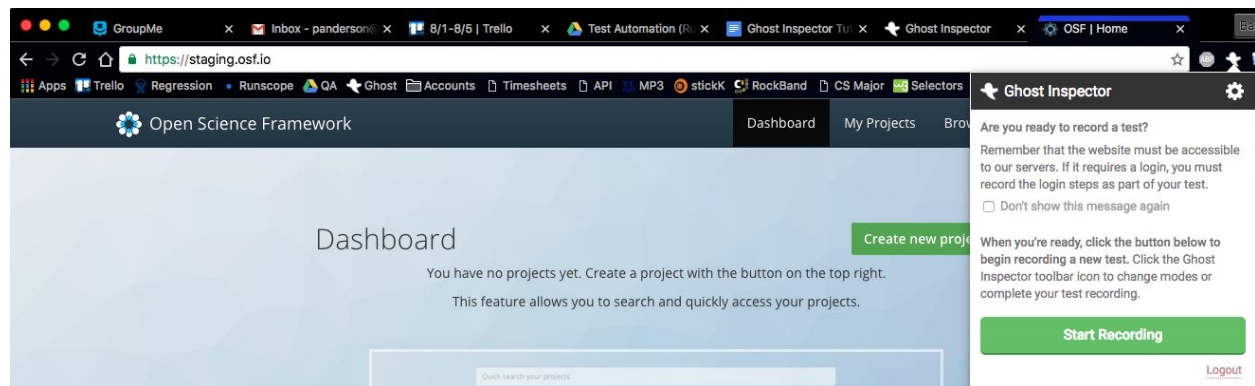
You have to go to staging.osf.io. Then you have to log in. Then you are taken to the dashboard, where you can create your project. Then you should ensure that your project was, in fact, created correctly.

I cannot stress how important it is for you to break up your tests into smaller tests. You don't want to make them *too* small, but you'll be surprised how small they can get. In our case, we are going to use a test that I have already created: a "Login" test.

Since we are going to use the “Login” test I have already created, we can assume that I have already logged in when we begin recording our tests.



Ok, so we are pretending that the test has already logged in for us, and has taken us to this page. Now we can start recording our tests.

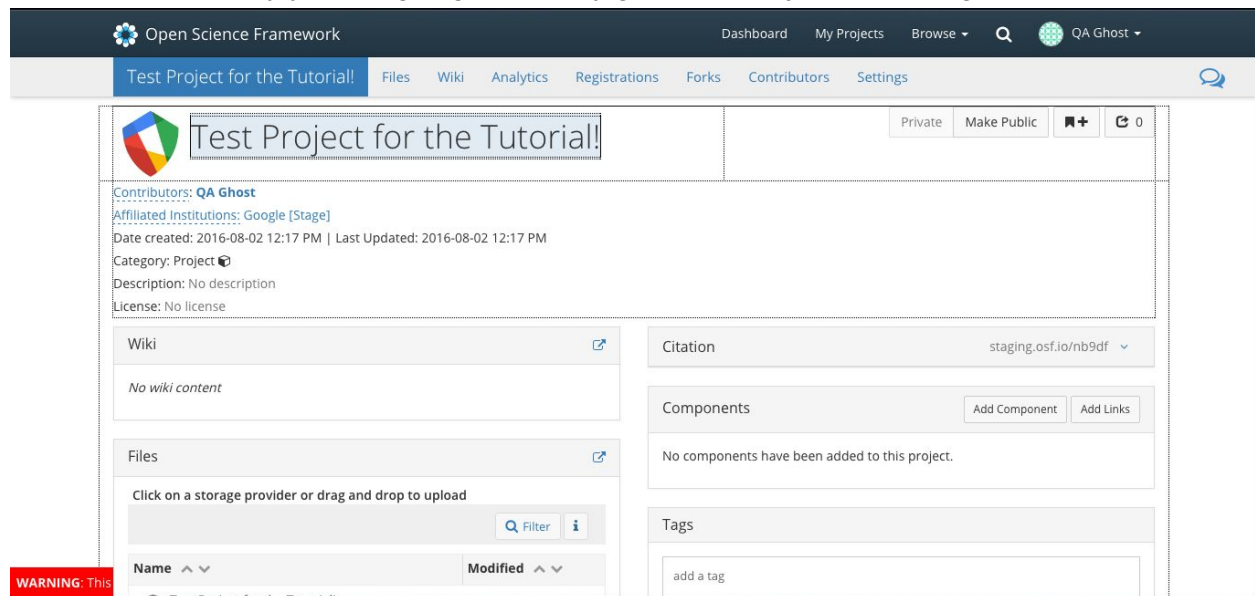


Click on the Ghost Inspector icon next to your toolbar. If you don't see it, go to your extension settings and make sure that it's turned on. Once you hit “Start Recording”, you will see this:



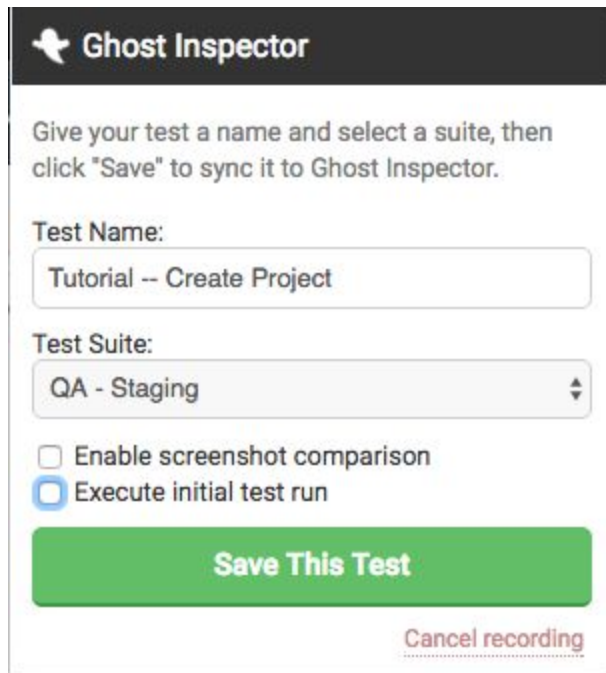
"Record Operations" is currently enabled. This means that whenever you click on anything or fill out any information, it will record it. The other option, "Make Assertions", makes it so that whenever you click on something, such as clicking on a button, it won't actually cause any changes (except for some rare cases), and instead only checks to make sure that that element exists in your test.

So now proceed to create your project as you normally would from the dashboard. It's only a few steps. Eventually you are going to actually get to the project's Title page:



This is where you will want to click on the little ghost icon on your browser bar and change your setting from "Record Operations" to "Make Assertions". Click on the title of the project. This will assert that the title should be whatever you named it; a decent indication during the test that you have reached the project's Title page, and it is the project that you just created.

So now, you have finished recording your operations, and can finish the recording by clicking on the ghost icon on your browser bar. Since you did this all on staging.osf.io, you want to save your test in “QA - Staging”.

A screenshot of the Ghost Inspector interface showing a dialog to save a test. The dialog has a dark header with the Ghost Inspector logo. Below the header, it says "Give your test a name and select a suite, then click 'Save' to sync it to Ghost Inspector." There are two input fields: "Test Name:" with the value "Tutorial -- Create Project" and "Test Suite:" with a dropdown menu showing "QA - Staging". Below these are two checkboxes: "Enable screenshot comparison" (unchecked) and "Execute initial test run" (checked). At the bottom is a large green button labeled "Save This Test" and a smaller red button labeled "Cancel recording".

Ghost Inspector

Give your test a name and select a suite, then click "Save" to sync it to Ghost Inspector.

Test Name:
Tutorial -- Create Project

Test Suite:
QA - Staging

☐ Enable screenshot comparison
☒ Execute initial test run

Save This Test

Cancel recording

Be sure to disable “Execute initial test run” and “Enable screenshot comparison”. You will almost always want to go to your test and check to make sure everything works before running it, as you will see very shortly.

Once you go to your test, try immediately running it as is. Notice that it fails instantly.

Latest Test Result

Tested on Aug 02, 2016 @ 12:18 PM — Test duration was 0:19
Start URL: <https://staging.osf.io/>

Test Steps

#1

Click on .row.m-t-1g

ELEMENT NOT FOUND

✖

URL: https://staging.osf.io/

Time: 12:18:58 PM

Edit

#2

Click on button.btn.btn-success.btn-success-high-contrast

#3

Click on input[name="projectName"]

#4

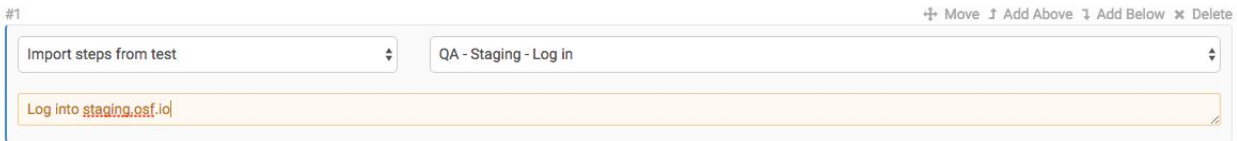
Assign Test Project for the Tutorial! into input[name="projectName"]

#5

Click on button[type="button"].btn.btn-success

As you might have predicted, this is because it tried to create a project from the Start URL while you were not logged in. You can see a screenshot of where it failed, and you will notice you are on the home screen, not on the dashboard, which is exactly what we would expect. We have the solution for this.

Edit your project. First of all, take out the Start URL so that it can inherit the start URL from the Suite Settings to make all of our tests consistent. See how the first step is failing? We want to add our “Log In” test above it. So click on the “Add Above” button and a new step will be created. Then fill it out so it looks like this:

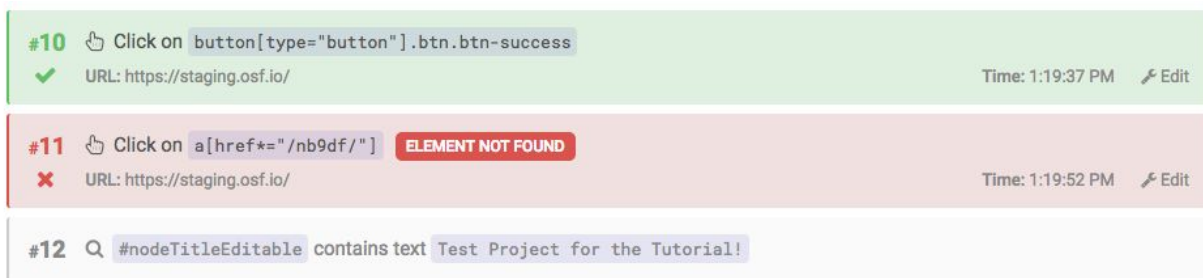


The left dropdown lets you choose an action. You can look through all the actions to see for yourself. We want the “Import steps from test” action. Then in the right dropdown, locate our “QA - Staging - Log in” test. This will make it so that our test will start with logging in by importing the steps from this test, and then execute the following steps we created.

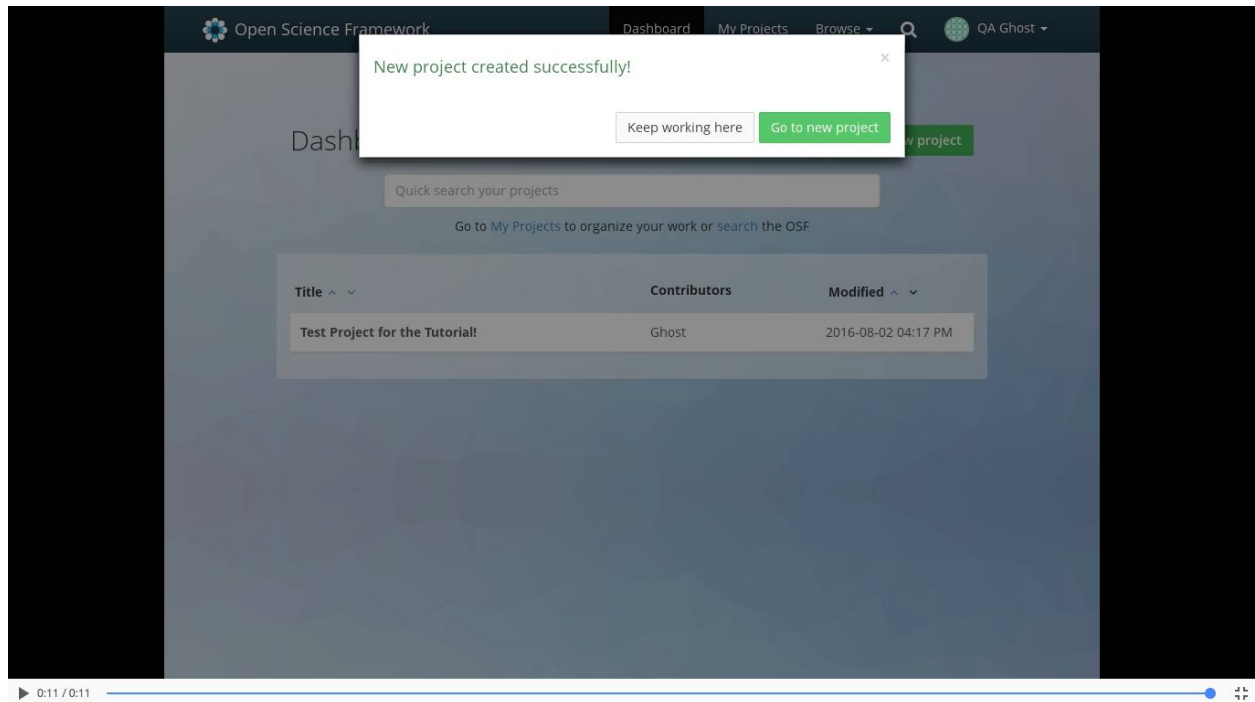
Ok! So it looks like we’re done, right? Go ahead and run the test again. Thanks for going through this with me! At this point, you’re totally ready to go!

WRONG.

The test failed. But why? Looks like it was step 11 for me:



Watch the video to see where it failed. Where did it fail? It failed when you click on the “Go to new project” button.



Why did this occur? The reason why this occurs is that this button is a link to your new project, which just so happens to have a unique GUID (for me it was nb9df). So when we ran our test again and created a different project, the element links us to a project with a different GUID.

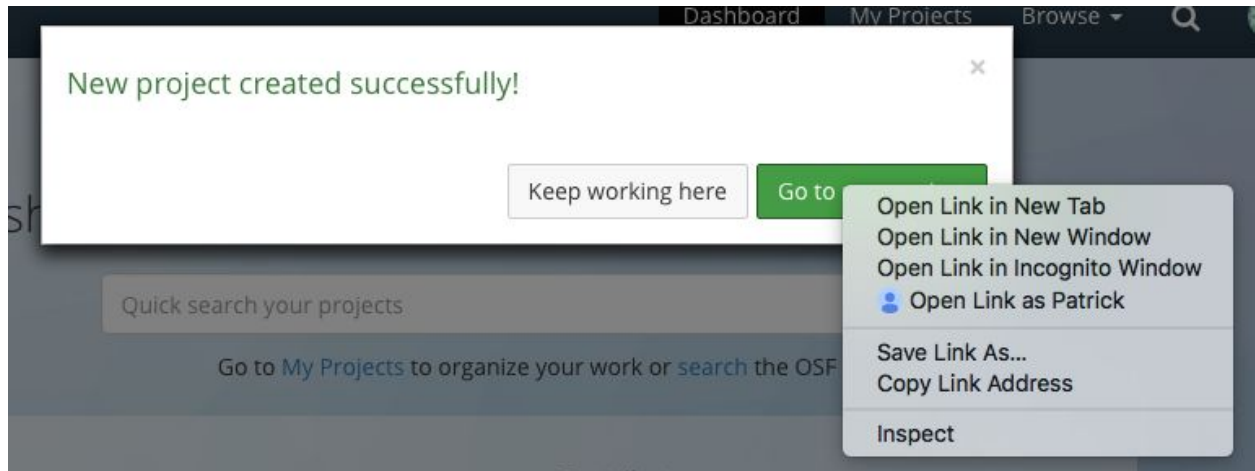
Hopefully this makes sense. Every time we run the test, whenever we click on the button “Go to new project”, it will take us to a new location, i.e. our new project, so the element is never the exact same.

Luckily there is a workaround to this, which took me a solid 10 hours to figure out.

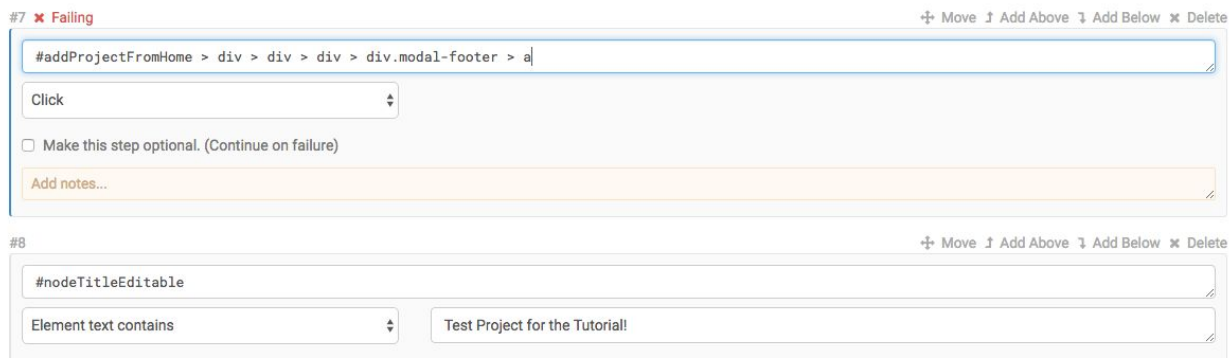
Essentially, instead of locating the element we want to click on in our test by its unique properties, we will instead locate it based on its location on the web page. For example, we know that the element is within the modal that popped up, which eliminates all of the buttons on the page down to the only two buttons on the modal.

Here’s the easy way to do this.

1. Go through the steps that you went through when making the project, and this time stop where the error occurred, i.e. when you reach the modal in the picture above.
2. Now this time, right click on the button that was causing the test to fail and click on the “Inspect” option:



3. A sidebar should appear with the element in the HTML that you wanted to inspect highlighted. Right click on this element and find the option “Copy” -> “Copy selector”. Ghost Inspector uses CSS selectors to identify what to click on and what to fill out on a web page ([here's](#) a brief summary of what they are), so what we are doing is copying the element to our clipboard.
4. Go back to your test and locate the step that was failing. Delete the text that Ghost Inspector filled out for us automatically when we were recording our steps and replace it with the text that we copied to our clipboard.



What this is essentially doing is making our selector as generic as possible (ours is literally only an “a”/href attribute), but specifying it by saying that it is in this EXACT location (designated by all of the “>” signs, which show that it is nested within each element). Therefore, it eliminates the specificity of an individual project!

Run your test again. At this point, mine worked, and I believe yours should, too. We have a few more loose ends we need to tie up.

Finishing Touches

First, it would be a good idea to look at each step in your test and add some notes to each step. You'll have a better understanding of what your test actually does, and it will be much more helpful for the future "you", say, 3 months from now when you have totally forgotten what you made, as well as anyone else reading your test.

Once you have read all of your steps and added notes to each one, we are going to take a brief look at how you can use variables in your tests.

Since we have been working in the QA - Staging suite, go to the QA - Staging suite settings. Then go to the "Variables" section of the settings. You should see something that looks like this:

The screenshot shows the 'QA - Staging' suite settings page. On the left is a 'Suite Settings' sidebar with options: Details, Schedule, Variables (selected), Default Settings, and Notifications. Below this is a red button labeled 'Delete This Suite...'. The main area is titled 'Variables' and contains a note: 'These variables will be available within all the tests in this suite.' Below this note is a table of variables:

username	osframeworktesting+ghost@gmail.com	<input type="checkbox"/>
password	c@us3c@us3c@us3	<input type="checkbox"/>
project_name	Test Project	<input type="checkbox"/>
component_name	Test Component	<input type="checkbox"/>
base_url	https://staging.osf.io/	<input type="checkbox"/>

Each row has a 'Make this value private' checkbox. At the bottom left is a blue button '+ Add Variable'. At the bottom right is a circular icon with three horizontal lines. In the top right corner of the settings area are 'Cancel' and 'Save Changes' buttons.

Take a look at what these variables represent. We have a variable called "username". Whenever we log in, instead of having to type out that entire email address, we can use the variable "username". When we run the test, Ghost Inspector will fill it in automatically. In our example, we'll replace the name we gave our project with the variable "project_name".

Go back to the test you created and Edit Steps. Locate the step where you assigned the project you were creating a name.

#5 ⛶ Move ⬆ Add Above ⬇ Add Below ✕ Delete

input[name="projectName"]

Assign ⬇ Test Project

☐ Make this step optional. (Continue on failure) Upload a file for use with a file input element.

☐ Make this value private. (Hide in display) Upload File...

Add notes...

Now replace it with this:

#5 ⛶ Move ⬆ Add Above ⬇ Add Below ✕ Delete

input[name="projectName"]

Assign ⬇ {{project_name}}

☐ Make this step optional. (Continue on failure) Upload a file for use with a file input element.

☐ Make this value private. (Hide in display) Upload File...

Add notes...

See what we did? It's going to do the exact same thing as before, but now we're using a variable instead of hardcoding a value into that step. For the most part, we do **not** want to be overloading our suite with random variables that will only be used in one or two tests. We want to be adding variables that we think will be used frequently, such as "username", "password", "project_name", "contributor_id", NOT "my_grandma's_birthday_that_I_am_using_in_one_test_to_check_user_settings".

Check your test one more time. Are there any other places where you want to replace a value with a variable?

I found one!

#8 ⛶ Move ⬆ Add Above ⬇ Add Below ✕ Delete

#nodeTitleEditable

Element text contains ⬇ {{project_name}}

☐ Make this step optional. (Continue on failure)

Add notes...

This step right here is the step we created to check if the name of our project matches the name we gave it when we were creating the project. Now both values should *always* match up.

OKAY. WE ARE ALMOST DONE HERE. It's been a ride. Here's the last part of our process: copying the test we created in "QA - Staging" over to "QA - Production".

Go to our Staging suite on the page where it displays all of the tests. Find the test you have made and select it like so:

QA - Staging ⚡ Bulk Actions ⚙ Settings ⌵ More ▶ Run All Tests

Center for Open Science — 10 Tests
Scheduled to run: [No Schedule](#)

This is the bucket for all of our staging tests. It has been formatted to resemble our QA - Staging bucket in Runscope, and to mirror our QA - Production bucket in Ghost Inspector. The numbered tests are complete tests that should be used to check function... [Show](#)

Test Name	Status	Disabled	Schedule	Last Run
<input type="checkbox"/> Log Out ▶ Run Test ⚙ Edit Steps ⚙ Settings	PASSING	DISABLED	No Schedule	07/29/16 2:49 PM
<input type="checkbox"/> Log in ▶ Run Test ⚙ Edit Steps ⚙ Settings	PASSING	DISABLED	No Schedule	07/29/16 2:37 PM
<input type="checkbox"/> Make a Project Public ▶ Run Test ⚙ Edit Steps ⚙ Settings	PASSING	DISABLED	No Schedule	07/29/16 2:55 PM
<input checked="" type="checkbox"/> Tutorial -- Create Project ▶ Run Test ⚙ Edit Steps ⚙ Settings	PASSING	DISABLED	Suite Schedule	08/02/16 1:22 PM

Next to “Settings”, there should now be some text you can click on called “Bulk Actions”. Click on it and click “Duplicate Selected Tests”. This will create a copy of the test you have been working on.

Now deselect your test and select the copy of your test. Bulk Actions should show up, and now you want to click on “Move Selected Tests to Suite...”

QA - Staging ⚡ Bulk Actions ⚙ Settings ⌵ More ▶ Run All Tests

Center for Open Science — 11 Tests
Scheduled to run: [No Schedule](#)

This is the bucket for all of our staging tests. It has been formatted to resemble our QA - Staging bucket in Runscope, and to mirror our QA - Production bucket in Ghost Inspector. The numbered tests are complete tests that should be used to check function... [Show](#)

Test Name	Status	Disabled	Schedule	Last Run
<input type="checkbox"/> Log in ▶ Run Test ⚙ Edit Steps ⚙ Settings	PASSING	DISABLED		
<input type="checkbox"/> Make a Project Public ▶ Run Test ⚙ Edit Steps ⚙ Settings	PASSING	DISABLED		2:55 PM
<input type="checkbox"/> Tutorial -- Create Project ▶ Run Test ⚙ Edit Steps ⚙ Settings	PASSING	DISABLED	Suite Schedule	08/02/16 1:22 PM
<input checked="" type="checkbox"/> Tutorial -- Create Project (Copy) ▶ Run Test ⚙ Edit Steps ⚙ Settings	UNKNOWN	DISABLED	Suite Schedule	N/A

▶ Run Selected Tests

📄 Duplicate Selected Tests

✓ Accept Screenshots as Baseline

⚡ Move Selected Tests to Suite...

✗ Delete Selected Tests...

A modal will pop up and ask you which suite you want this to be moved to. You want to select “QA - Production”. The test you copied over is now in the “QA - Production” suite. Locate it there.

Now, the last thing you want to do is go to the test and edit the steps. There is one more thing we need to change. Remember the Login test we imported? We want to replace the Login test that is in the Staging suite with the Login test that is in the Production suite:

Edit Steps

Test steps allow you to perform actions within the browser. They are executed sequentially.

Start URL

http://staging.company.com (Leave blank to use suite setting)

#1 ⛶ Move ⬆ Add Above ⬇ Add Below ✕ Delete

Import steps from test QA - Production - Log in

Log into staging.osf.io

#2 ⛶ Move ⬆ Add Above ⬇ Add Below ✕ Delete

.row.m-t-lg

Click

See that? Now everything in the Production suite is enclosed in the Production suite, and everything in the Staging suite is enclosed in the Staging suite. There is no crossover.

That, is finally it. Hopefully you now know the basics of what you might be doing in the future. I recommend looking at some of the building block tests to see more of what you can do with Ghost Inspector. Click around to make sure you know what all of the buttons do and what each option means. I walked you through this example; now you can try making a test session on your own!

-Patrick

Additional Notes

(As a side note, it would be a good idea to delete the tests you made in the tutorial once you feel comfortable enough to begin working on the tests we need for regression testing.)

Some additional notes:

1. You will want to look at the Create Project building block and understand how it goes about getting the project's GUID. This is incredibly important for future tests.

- For some reason, from time to time Ghost Inspector automatically autofills these forms in these pictures with your Ghost Inspector username and password. THIS IS A MAJOR PROBLEM. If you ever see anything in these fields, delete them immediately. What it does is that it makes it so that it logs you in automatically. Since every test begins with logging in, they will all fail. Alternatively, make your browser forget your Ghost Inspector login credentials so this does not happen.

Dashboard > QA - Staging > Suite Settings

QA - Staging

Cancel Save Changes

Suite Settings

- Details
- Schedule
- Variables
- Default Settings**
- Notifications

Delete This Suite...

Default Test Settings

These settings can be overridden at the test level.

URLs

Start URL

Browser

Browser Version

Custom User Agent

Authentication

HTTP Auth Username

HTTP Auth Password

Screenshots

Screen Size

Dashboard > QA - Staging > Create a Component > Test Settings

Create a Component

Cancel Save Changes

Test Settings

- Details
- Schedule
- Browser Access**
- Step Timing
- Display Options
- Notifications

Delete This Test...

Browser Access

URLs

Start URL

Browser

Browser Version

Custom User Agent

Authentication

HTTP Auth Username

HTTP Auth Password

- These are the step timing settings I typically use:

Dashboard > QA - Staging > Create a Component > Test Settings

Create a Component

Cancel

Save Changes

Test Settings

Details

Schedule

Browser Access

Step Timing

Display Options

Notifications

Delete This Test...

Step Timing

Step Delay

0 seconds10 seconds

The test will pause for 0 seconds between each step so the browser has time to process events.

Element Timeout

15 seconds60 seconds

The test will spend up to 15 seconds attempting to locate an element before failing.

XHR Request Timeout

3 seconds30 seconds

The test will spend up to 3 seconds waiting for XHR requests to complete before continuing on to the next step.

Final Screenshot Delay

0 seconds30 seconds

The test will pause for 0 seconds at the end before taking the final screenshot.

Test Auto-Retry

☐ Automatically retry the test if it fails (when the last test run passed)

If something fails and it seems like it is because the page took too long to load and so Ghost Inspector stopped, try increasing the Step Delay time.

Resources

- Ghost Inspector: <https://app.ghostinspector.com/>
- Brainstorming automated tests Google Doc: https://docs.google.com/document/d/1Bdan1ER9r4nZfdPGkNJcX9o0N1EW_JBpIEgnOPX4DP8/edit#heading=h.ewxpijzgns1c
- Ghost Inspector Chrome Extension: <https://chrome.google.com/webstore/detail/ghost-inspector-automated/aicdiabnghjnejfempeinmnphllefehch?hl=en-US>
- Information about CSS Selectors: http://www.w3schools.com/cssref/css_selectors.asp

16